Szakdolgozat

Szilvási László

Debrecen

2009

Debreceni Egyetem

Informatika Kar

Képfeldolgozáson alapuló mozgásvizsgálat

Témavezető:

Dr. Szabó István
egyetemi docens
Szilárdtest Fizika Tanszék

Készítette:

Szilvási László
Mérnök Informatikus

Debrecen 2009

TARTALOMJEGYZÉK

1. BEVEZETÉS

A szakdolgozat témája a Debreceni Egyetem Természettudományi és Technológiai Kar Szilárdtest Fizika Tanszékén készített program és a kapcsolódó fejlesztői környezet bemutatása. A program feladata, hogy képes legyen gyorsulásmérőket ellenőrizni, hitelesíteni. A programot a National Instruments által kifejlesztett fejlesztőkörnyezetekben készítettem el.

Ahhoz, hogy ezt a témát választottam, nagyban hozzájárult, hogy ezeket a programozási nyelveket használtuk az órákon. Az egyetemi tanulmányok alatt is nagy lehetőségeket láttam a LabVIEW nyelv használatában. A képfeldolgozáson alapuló mozgásvizsgálathoz ideális LabVIEW nyelv és a hozzátartozó kiegészítő lehetőségek keltették fel az érdeklődésemet. A program megírása során nem csalódtam. Az alkalmazott programozási nyelvek izgalmas lehetőségeket jelentettek, új ismeretekre is szert tettem.

A dolgozatom témájához szorosan kapcsolódik a képfeldolgozás melyet az első fejezetben ismertetek. A technikai fejlődéssel ma már viszonylag könnyen hozzáférhető digitális eszközök segítségével a képek rögzítése és digitalizálása egyszerűen megoldható. A dolgozat tárgyát képező mozgásvizsgálatához szükséges képrögzítés is egy digitális kamera használatával valósult meg, mely kiváló minőségű videójával tökéletes input adatot szolgáltatott programom számára.

A szakdolgozat következő fejezetében a fejlesztői környezetet mutatom be, részletezve a programozási nyelvek alapjait, sajátosságait, az alkalmazott főbb függvényeket. Ebben a fejezetben külön foglalkozom a talán legnagyobb segítséget nyújtó JKI State Machine alkalmazhatóságával. Ennek használata újszerű volt számomra tekintettel arra, hogy az egyetemi tanulmányok alatt ezzel a függvénnyel nem ismerkedtünk meg.

A dolgozat keretében ismertetem a mérés fizikai környezetét, a méréshez használt eszközt és magát a folyamatot.

A megalkotott szoftver célja, hogy képfeldolgozás segítségével megvizsgálja a rezgőmozgást, a kapott adatokat feldolgozza és azokból meghatározza gyorsulás nagyságát.
A program működtetése keretében megállapításra kerül, hogy futtatása egyszerű, mindössze négy ablak használatából áll, melyek könnyen kezelhetőek, áttekinthetőek, egyértelműek. Ez köszönhető a programnyelvek és a kiegészítő függvények adta lehetőségeknek.
A következő fejezetben ábrák és képek segítségével mutatom be a program felépítését. A blokk diagram egyes részeit bemutató képeken megfigyelhetjük, tanulmányozhatjuk a program működésének fontos részeit.
A tárgyalási rész zárásaként a mérési eredményekkel foglalkozom, melynek keretében bemutatom a program és a gyorsulásmérő eszköz által végzett összefoglaló adatokat. Ezeket táblázaton és grafikonos formában is ábrázolom.

A szakdolgozatot és dolgozat alapjául szolgáló programot nagy lelkesedéssel írtam meg mert a tanulmányaink során viszonylag keveset használt programozási nyelvek és az azokat kiegészítő VI-ok és függvények megismerése és használata kihívást jelentett számomra. A programozás során a legérdekesebb feladat a képkalibrálással járó akadályok leküzdése volt, melyeket sikerrel oldottam meg.

2. A KÉPFELDOLGOZÁSRÓL

[1]. A képfeldolgozás gyorsan fejlődő terület, melyet elsősorban talán a digitális fényképezés gyors elterjedése és mindennapi használata segít. A számítógéppel történő képfeldolgozás célja és lényege, hogy a vizsgálni kívánt képi információt – megfelelő átalakítás útján - a számítógép segítségével feldolgozzuk és kiértékeljük. Ennek során első fázisban a valós világ tárgyainak képét számítógéppel kezelhető adatokká kell átalakítani. Ezt a transzformálást nevezzük digitalizálásnak. A digitalizálás alapvetően három lépésben végezhető el:

- A ***leképezés*** során a képalkotó eszköz pl. a kamera vagy a fényképezőgép – a tárgyakról érkező elektromágneses jelekből alkot képet. Az eljárás során a három dimenziós térben lévő tárgyakról kétdimenziós képek jönnek létre.
- A második fázisban a ***mintavételezés*** történik, amikor is a már elkészült kétdimenziós képekhez képpontonként (pixelenként) számokat rendelünk. A mintavételezés a fő tényező, ami meghatározza egy kép térbeli felbontását. Alapjában véve a térbeli felbontás a legkisebb felismerhető részlet egy képen. A térbeli felbontás mértékegységeként jellemzően a *dpi*-t használják, ami a dot per inch rövidítése, de az inch helyett használhatunk milimétert, centimétert, vagy más, nekünk tetsző mértékegységet, természetesen ekkor az értékét át kell váltanunk.
- Az úgynevezett ***kvantálási*** művelet a harmadik fázis, amikor a mintavételezett képet alkotó tetszőleges értékű számértékekhez a megengedett világosságkód értékek valamelyikét rendeli. A minták értékei egy folytonos szürkeségi érték tartományt fognak át. Ezért ahhoz, hogy egy digitális függvényt alkossunk, a szürkeségi értékeket konvertálni (kvantálni) kell diszkrét mennyiségekre. A képfüggvény folytonos fényességértékei és a digitális megfelelőik közötti átalakítást nevezzük kvantálásnak. Eredménye egy képfüggvény, aminek véges sok, diszkrét értéke van, vagyis diszkrét értékű képpontokat kapunk, amelyeket általában egész számokkal reprezentálunk. A kvantálási szintek számának elegendőnek kell lennie ahhoz, hogy érzékelhetők legyenek a finom(an árnyalt) színátmenetek, részletek a képen.

folytonos képfüggvény → kvantált képfüggvény

kvantáló

vagy

analóg → digitális (A/D)

vagy

valós → egész

A digitalizálás eredményeként kapott számhalmazt, a digitális képet a számítógép már tudja kezelni, és a programon múlik, hogy ebből a számhalmazból egy adott probléma esetén, mit és hogyan tudunk kiolvasni. Színes kép esetében a mintavételezés és a kvantálás szín-összetevőnként, egymástól függetlenül valósul meg.

1. ábra Egy képfeldolgozási lehetőség

3. FEJLESZTŐ KÖRNYEZET

3.1.A LabVIEW programozási nyelv

2. ábra LabVIEW ikon

[2]. A LabVIEW (Laboratory Virtual Instrument Engineering Workbench) egy grafikus programozói nyelv, amelynek alkalmazási területe a virtuális méréstechnika és automatizálás. A LabVIEW általános mérnöki eszközként használható mérési, tesztelési és szabályozási feladatokhoz, valamint beágyazott rendszerek fejlesztéséhez. A LabVIEW igen széles eszközkészlettel rendelkezik az adatgyűjtés, adatelemzés, megjelenítés és adattárolás területén, valamint számos eszközt tartalmaz a hibakeresés segítésére. A LabVIEW programokat virtuális műszereknek, vagy rövidebben VI (Virtual Instrument) – oknak nevezzük, mert megjelenésükben és működésükben a fizikai műszereket utánozzák.

A LabVIEW VI-ok az három komponensből állnak:

- az előlapból (front panel),
- blokk diagramból (block diagram)
- az ikon és csatlakozó panelből (icon and connector panel).

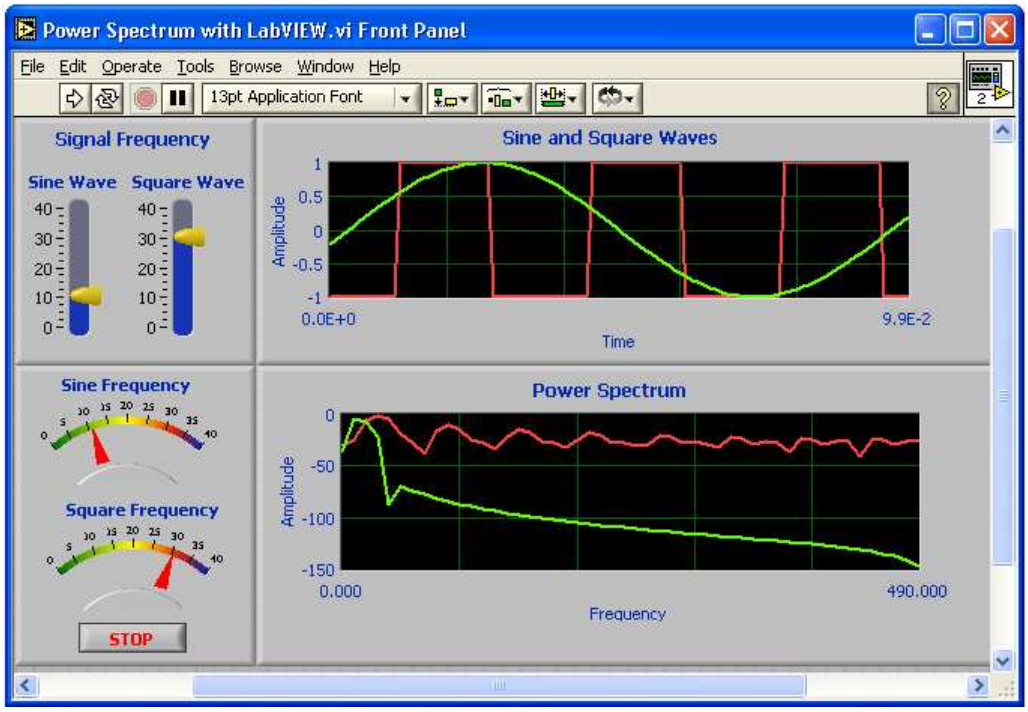
A LabVIEW VI-ok használatával tesztelő és mérő, adatgyűjtő, műszervezérlő, adatnaplózó és méréselemző alkalmazások készíthetők. A program a National Instruments terméke. 1986-ban Apple Macintoshra fejlesztették ki, de futtatható egyéb platformokon is.(Pl. Windows, Unix különböző verziói, Linux) A LabVIEW kompatibilis más programokkal. (Pl. Matlab, MathWorks, NI MMATRIXx.)

3.1.1. Front panel

LabVIEW-ban vezérlők (control) és kijelzők (indicator) segítségével készíthetjük el a felhasználói interfészt vagy más néven az előlapot (front panel).

- Az itt elhelyezhető elemek:
  - Vezérlő elemek
    - ezek a program bemenetei; értékük manuálisan változtatható A vezérlők közé tartoznak a forgatógombok (knob), a nyomógombok (push button), a tárcsák (dial) és más input eszközök
  - Kijelzők
    - program kimenetei; értéküket a program futása közben nyerik. A kijelzők közé taroznak a grafikonok (graph), *****

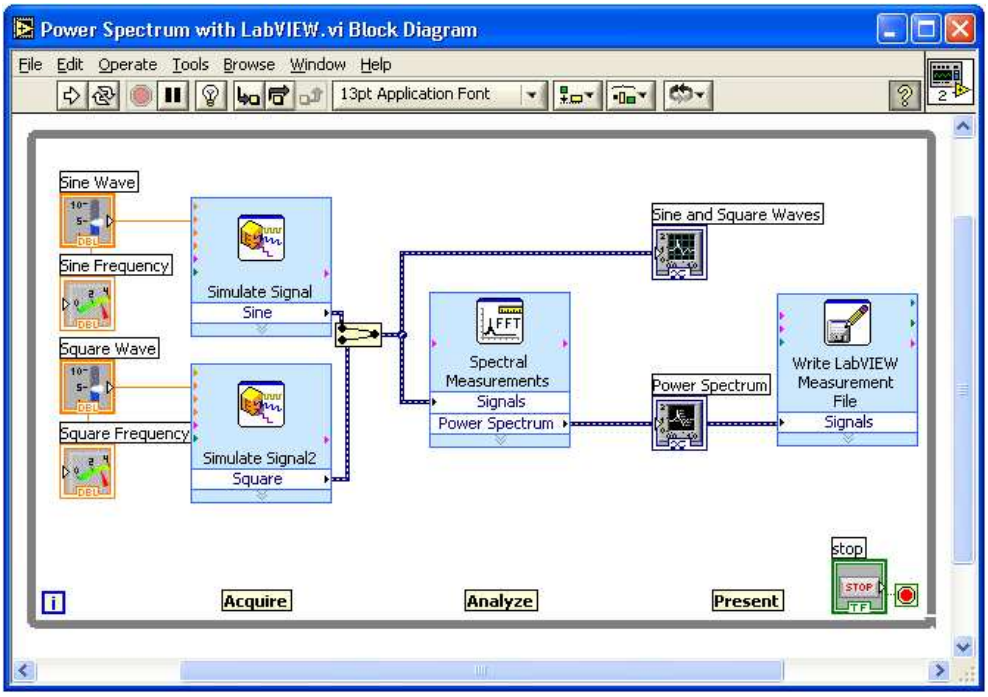
Attól függően, hogy milyen típusú adatot szeretnék manipulálni vagy megjeleníteni, úgy változik a vezérlő és a kijelző is. Az adat belső reprezentációjától függően a LabVIEW automatikusan felismeri, hogy milyen típusú adatot szeretnék megjeleníteni. Tömb adatszerkezet esetén is megtörténik az automatikus típuskonverzió..



3. ábra Front panel

3.1.2. Block Diagram

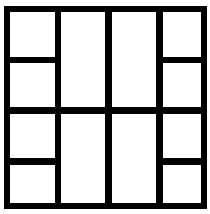
Tágabb értelemben a blokk diagram maga a fejlesztői környezet. Míg az előlapi panel a külső megjelenésért felelős, a blokk diagram az adatfolyam programozásért felel. A be- és kimenetek között a tényleges adatáramlás vezetékeken át, a program által definiált módon történik. A vezetékek csomópontokban találkoznak. A csomópontok funkciójuk szerint lehetnek függvények, struktúrák és SubVI-ok. A SubVI-ok felhasználhatósága igen széleskörű. A fejlesztő saját maga definiálhat egy SubVI-ban függvényeket, melyek tetszőleges mélységben egymásba ágyazhatók. Az ismétlődő kódrészlet kiemelése mellett a rekurzió is könnyebben megvalósítható. Mindezek mellett javul a program blokk diagrambeli struktúrájának az áttekinthetősége. A LabVIEW lehetővé teszi a megírt program működésének, az adatáramlásnak lépésenkénti vizsgálatát, így a virtuálisan megépített hardver elemek könnyen tesztelhetők. A felhasználói interfész elkészítése után VI-ok és struktúrák felhasználásával ahhoz egy kódot rendelhetünk hogy az előlapi objektumokat vezéreljük. A blokk diagram ezt a kódot tartalmazza. Bizonyos tekintetben a blokk diagram egy folyamatábrára hasonlít.



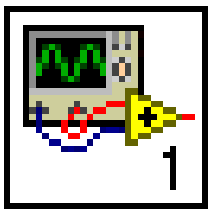
4. ábra Block diagram

3.1.3. Icon and connector pane

[3]. Segítségükkel úgynevezett subVI –ok hozhatók létre. Ezek egyszerűbbé, átláthatóbbá tehetik programunkat. A konnektor panelon kijelölhetőek a be és kimenetek és azok típusai. A VI ikon grafikai megjelenése a .VI-nak. Ez tartalmazhat szöveget, képet és a kettő kombinációját.
Ha egy VI-t mint subVI-t használunk az ikon azonosítja a blokk diagramon.



5. ábra Konnektor panel



6. ábra Ikon

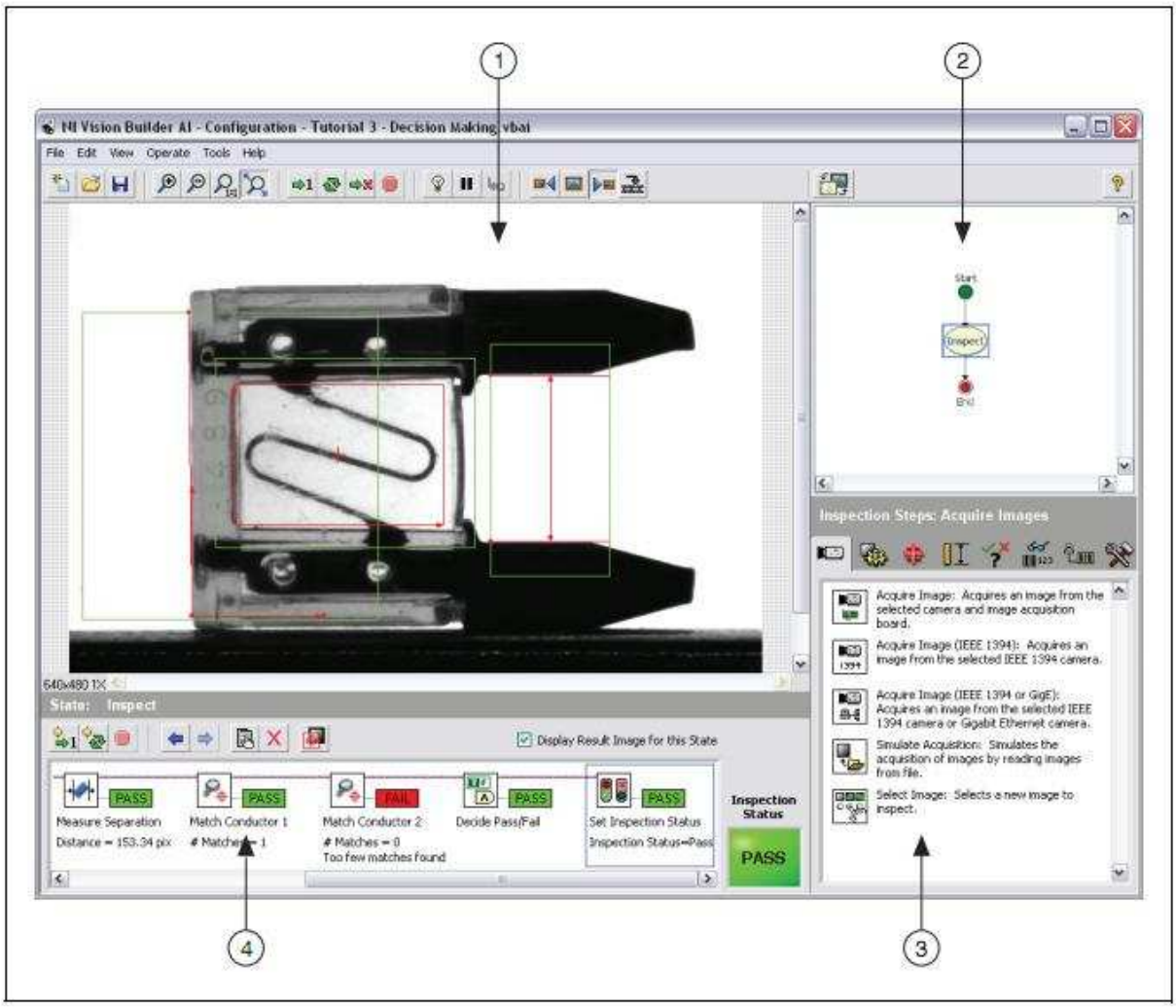
3.2.A LabVIEW Vision

A LabVIEW egy kiegészítő modulja a Vision, ami a képfeldolgozással kapcsolatos függvényeket tartalmaz. A képfeldolgozás sikeres végrehajtásához a LabVIEW jól megszokott jelfeldolgozási rendszerét kiterjesztették hardware és software oldalon..
A LabVIEW -n belül az IMAQ programcsomagban találhatóak a speciális képfeldolgozást biztosító függvények. Az IMAQ részét is képező Vision Assistant program segíti a képfeldolgozási feladatok interaktív konfigurálását. A Vision Builder program, - amely használja a Vision Assistant szolgáltatásait is - .többek között a gyártási - minőségellenőrzési feladatok interaktív programozását is segíti. A varázslókkal összeállított képfeldolgozási megoldásokból LabVIEW program generálható, amely így könnyen továbbfejleszthető.
Hardware oldalon a megfelelő driverek és csatolókártyák segítségével lehetőség van analóg, USB, firewire és kameralink csatlakozású kamerák jeleinek fogadására

3.2.1. Vision Builder AI

[4]. A LabVIEW Vision Builder AI (Builder for Automated Inspection) a National Instruments által kifejlesztett fejlesztői környezet mely rendkívül megkönnyíti a képfeldolgozást, a képfeldolgozással kapcsolatos műveleteket. A szoftver használata nagyon egyszerű, minimális programozói tudással rendelkezők számára is biztosítja a gyors programozást.

Vision Builder AI konfigurációs felület



7. ábra Konfigurációs felület

1. Főablak
2. Áttekintő ablak
3. Ellenőrzési lépések panel
4. Állapotkonfigurációs ablak

A Vision Builder AI megnyitása után láthatjuk hogy a konfigurációs felület négy részre osztható. A főablak, az áttekintő ablak, a függvények panelje és az állapot konfigurációs ablak.

1. A főablak: ez jeleníti meg a feldolgozás alatt álló képet. Itt lehet kijelölni például ha a képnek csak egy részét akarjuk vizsgálni, itt konfigurálhatóak egyes függvényeknek a paraméterei és itt lehet létrehozni vagy módosítani egy állapot diagramot.

2. Az áttekintő ablak: ez kis ikonokkal mutatja nekünk a képfeldolgozás folyamatábráját vagy a folyamat egy állapotát.

3. Ellenőrzési lépések panel: Itt találhatóak a függvények melyeket használhatunk a képfeldolgozás során, rövid leírással együtt. A legtöbb függvényre kattintva felugrik a függvény saját oldala ahol a beállítási lehetőségeket láthatjuk.

4. Állapotkonfigurációs ablak: A függvények azon listáját mutatja, amelyeket a konfigurálás során kiválasztottunk, és amelyek a program futása során használatra kerülnek.

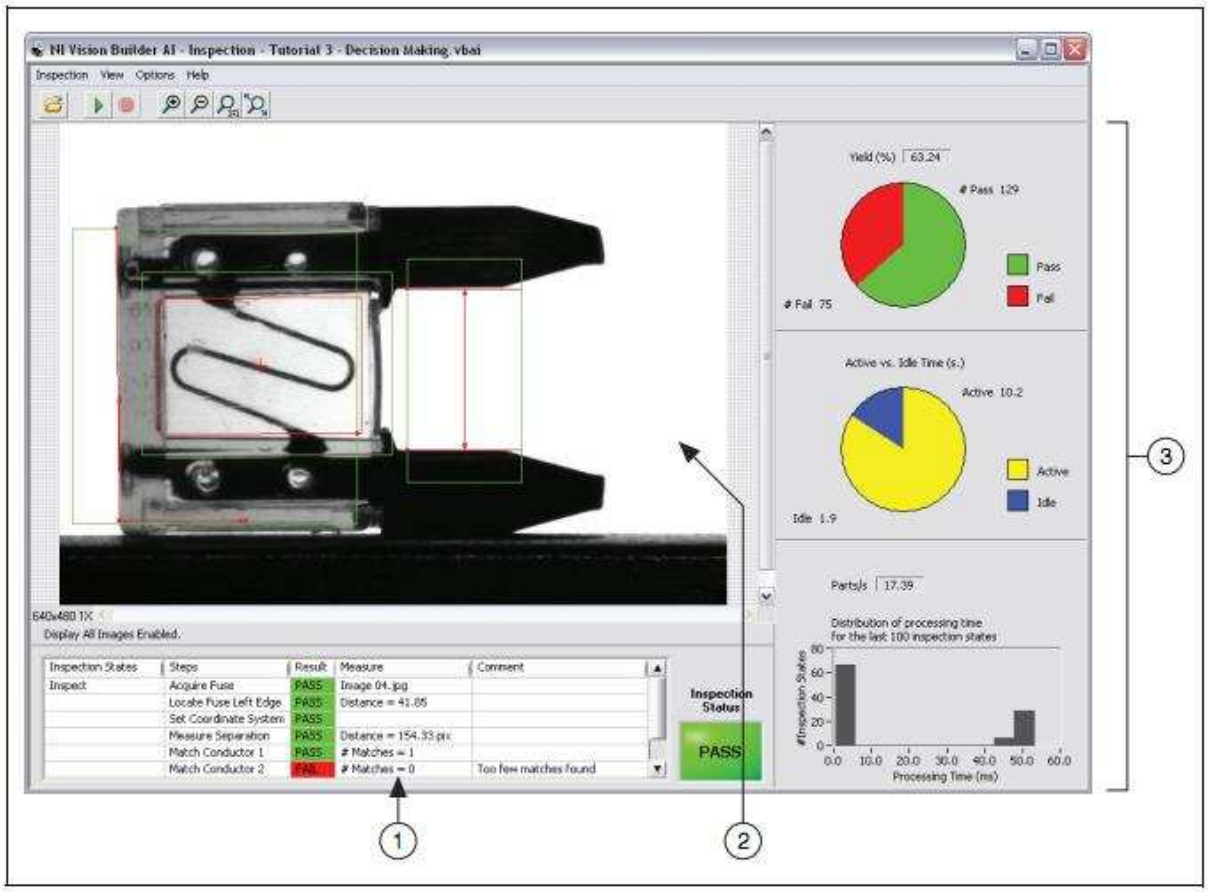
Vision Builder AI ellenőrzési felület

A ellenőrzési felület is, akár a konfigurációs felület, több részből tevődik össze. Ez a három rész az eredmények panel, ellenőrzési statisztika panel és megjelenítő ablak.

1. Eredmény panel: Kilistázza a folyamatban résztvevő függvényeket név szerint sorba rakva és mindegyikhez mellérendelve a függvény típusát, az eredményt (SIKERES, NEM SIKERES), a mérés típusát, egy megjegyzést arról hogyha a függvényt nem sikerült végrehajtani és egy folyamat státuszt sikeresen lefutott-e a folyamat egésze.

2. Megjelenítő ablak: A megjelenítő ablakban láthatjuk a feldolgozás alatt álló képet.

3. Folyamat statisztika: Három kijelzőt tartalmaz, az egyik a SIKERES, NEM SIKERES arányt rajzolja ki egy körgrafikonon, a második az aktív és az alvó állapotot ábrázolja szintén egy körgrafikonon és a harmadikon a folyamat lefutásának idejét láthatjuk.



8. ábra Vizsgálatai felület

1. Eredmények panel 2. Megjelenítő ablak 3. Ellenőrzési statisztika

Egy folyamat futtatása

A Vision Builder AI engedi hogy mind a konfigurációs mind pedig a ellenőrzési felületen futathassunk folyamatokat. A következő részben bemutatom a futatással kapcsolatos lehetőségeket mindkét felületen.

Konfigurációs felület

A konfigurációs felületről történő futtatás jól jöhet a tesztelésnél és hibakeresésnél. Az utóbbi alkalmazása során igen hasznosak lehetnek a „Highlight Execution” , „Pause” és a „Single Step” gombok. Az alábbi táblázat a további lehetőségeket taglalja.

Gombok	Név	Leírás
	**Run Inspection Once**	Egyszer futtatja a folyamatot
	**Run Inspection in Loop**	Folyamatosan futtatja a folyamatot
	**Run Inspection Until Failure**	Addig futtatja a folyamatot amíg a folyamat státusz változó nem ad HAMIS értéket
—	**Run Inspection Multiple Times**	Futtatja a folyamatot bizonyos számszor
	**Stop Inspection**	Leállítja a folyamatot
	**Highlight Execution**	Megvilágosításos futtatás. Ha be van kapcsolva megvilágítja az éppen futó részt a folyamatban. Ilyenkor a gombon az izzó sárga színűre vált
	**Pause**	Szünteleti vagy folytatja a folyamat futtatását. Ha a gomb piros, akkor a folyamot szüntel.
	**Single Step**	Csak akkor működik ha a folyamat szüneteltetve van. Egy művelettel továbbengedi a folyamtot.

9. ábra Nyomógombok és leírásuk

Ellenőrzési felület

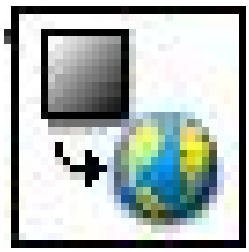
Amennyiben a vizsgálatai felületen kívánunk futtatni először át kell váltanunk a konfigurációs felületről. Ezt a „File»Switch to Inspection Interface” menüponttal tehetjük meg. Miután megnyitottunk egy folyamatot és átváltottunk ellenőrzési felületre a „Start Inspection”  gombra kattintva elindíthatjuk azt. A Vision Builder AI elindítja a folyamatot és frissíti mindhárom részt a ellenőrzési felületen a legújabb adatokkal. Alapértelmezetten mutatja a főablakban a képen történő feldolgozást, mérést, vizsgálatot. Ezt a megjelenítést megváltoztathatjuk a „View” menüpontban, továbbá a kép nagyítását is állíthatjuk az

„Options” menüben. Érdemes a vizsgálat során figyelemmel lenni a folyamat statisztika paneljében megjelenő adatokra mert ezek segíthetik eldönteni milyen hatékonyan fut a folyamat. Ezen túl ezek az adatok segíthetnek behangolni azt is, hogy gyorsabban futhasson a folyamat. A „Stop Inspection”  gombra kattintva megállíthatjuk a folyamatot.

3.3.A Vision Builder AI-ban használt főbb függvények

[5]. A Vision Builder AI program nagyban könnyítette munkámat. Ezen program használatával készítettem egy alprogramot, amelyet átgenerálva subVI-ként használtam fel a képek kalibrálására a fő programomban.

3.3.1. Calibrate Image

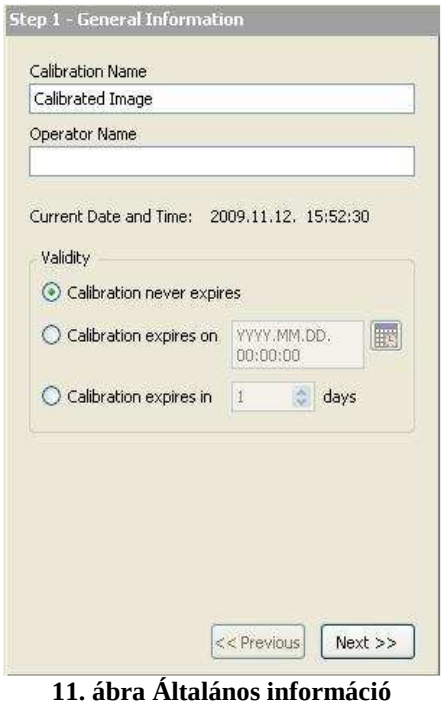


10. ábra Calibrate Image ikon

Ezzel a függvénnyel kalibrálhatunk egy képet a való világ egységeinek megfelelően, mint ahogy a programban is, a calibrated image-ben is meg van adva hogy ami azon a képen X pixel az a valóságban hány centiméter.

Első lépésként megadhatunk egy külön nevet a „Step Name” felirat alatti vezérlőben.

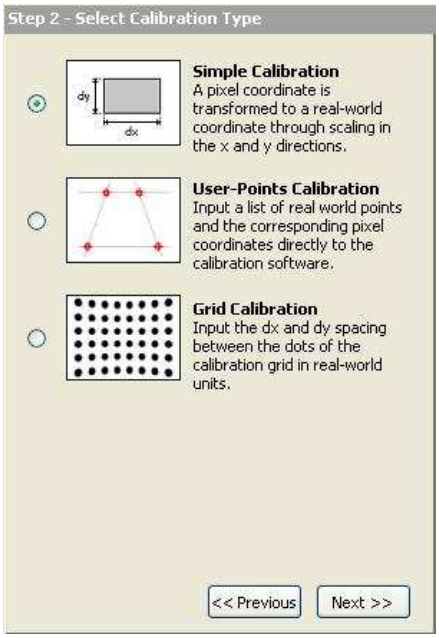
Ezután a „New calibration” gombra kattintva meg kell adni a kalibrált kép nevét és érvényességét, ezt láthatjuk a 11. ábrán.



11. ábra Általános információ

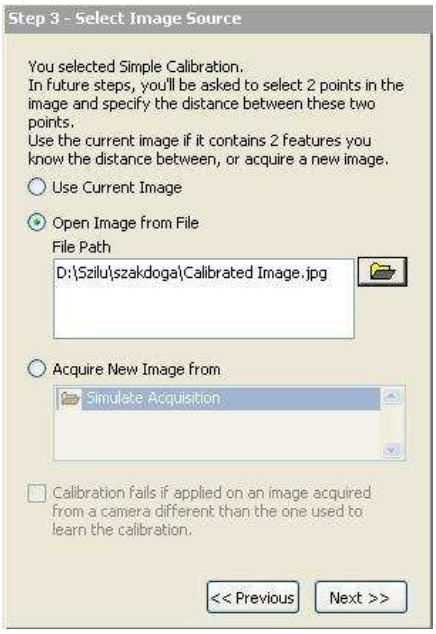
A „Next” gomb lenyomása utána választhatjuk ki hogy milyen fajta kalibrálást szeretnénk végezni rajta.

.



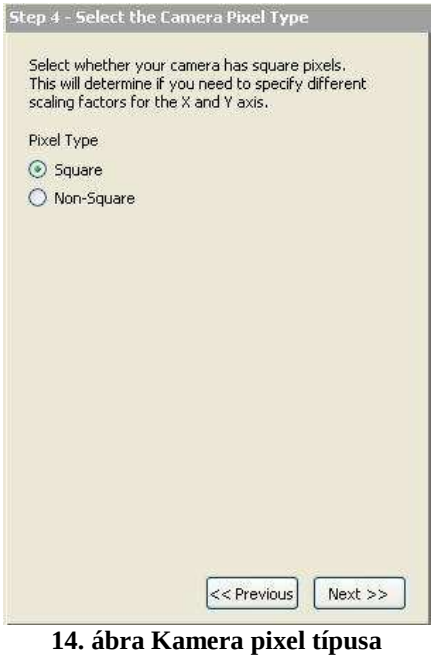
12. ábra A kalibrálás típusa

A programban szimpla kalibrációt alkalmaztam mivel egyszerűen a két pixel távolságára vagyunk kíváncsiak.



13. ábra Kép megnyitása

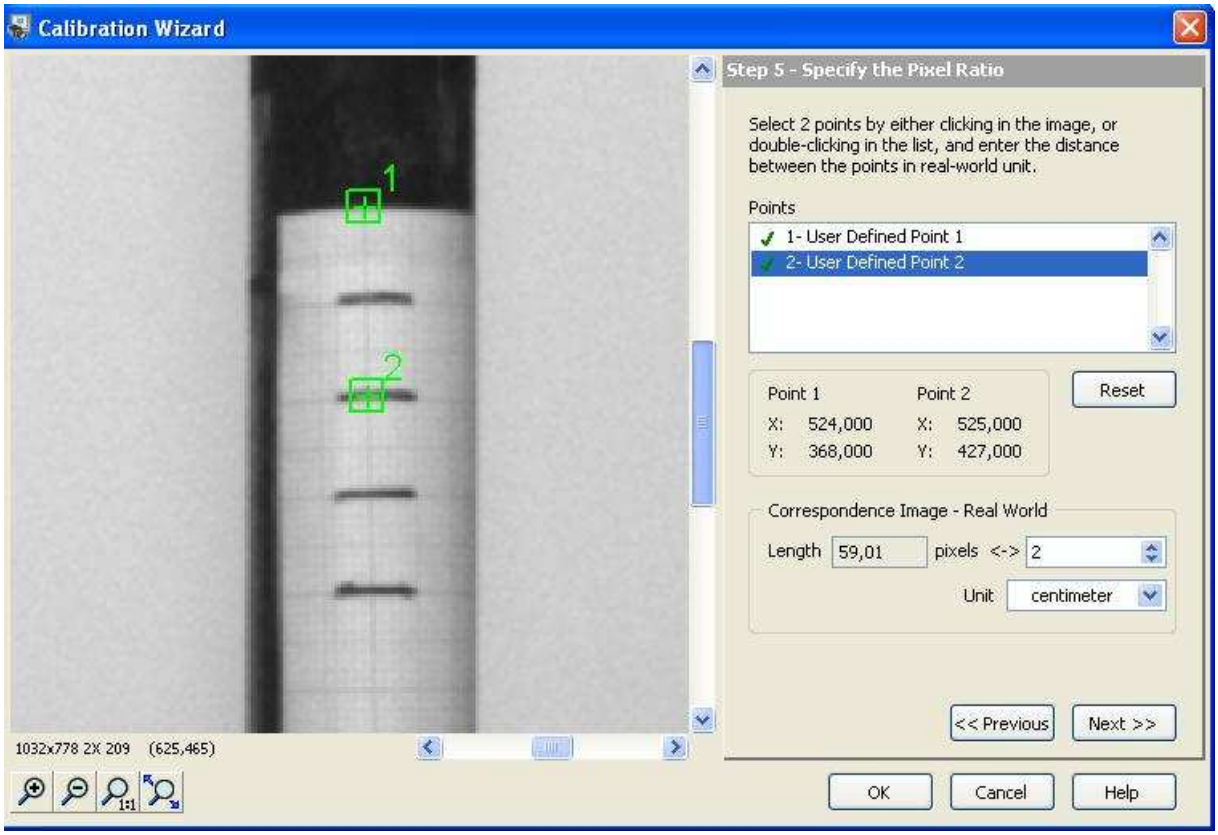
Ezen lépésnél választhatjuk ki hogy melyik képen határozzuk meg majd azt a két pontot amelyeknek távolságát akarjuk megtudni.



14. ábra Kamera pixel típusa

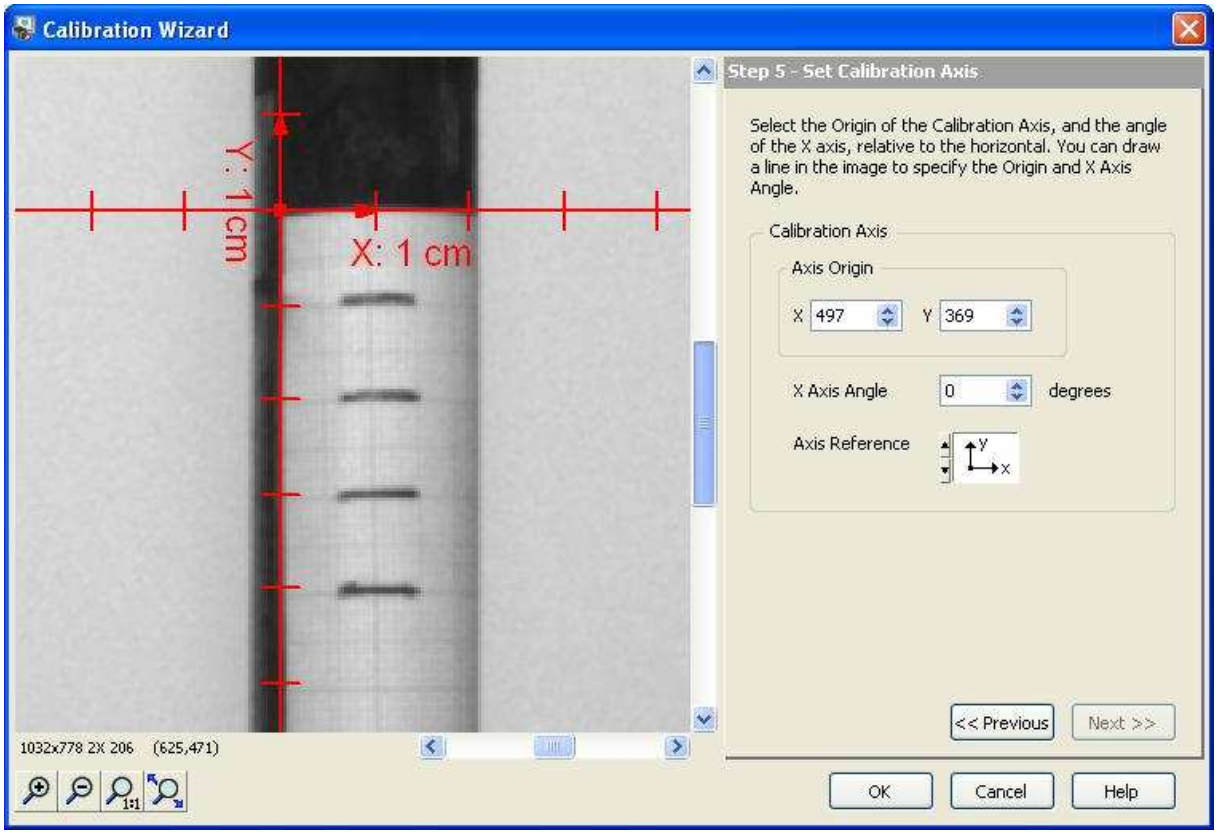
A következő lépésnél a kamera pixel típusát tudjuk beállítani.

Elérkeztünk ahhoz a lépéshez ami a legfontosabb, mikor is bejelöljük a képen a két pontot. Ahogy a következő képen jól látható a két általunk megjelölt pontnak kiírja a koordinátáját (Point 1, Point 2) alatta pedig a Lenght kijelzőben kiírja a két pont közti távolságot pixelben. Mellette láthatjuk az Unit vezérlőt ahol az egységet kell megadnunk. Ahogy 15. ábra is mutatja, úgy sikerült jelen esetben beállítani, hogy ami a képen 59,01 pixel távolság az a valóságban 2 centiméter.



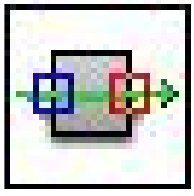
15. ábra Pontok bejelölése

Lehetőséget biztosít a program arra is, hogy az utolsó lépésben koordinátarendszert rakhassunk a képünkre, egy vonalhúzás segítségével (ez a vonal fogja jelenti az X tengelyt) vagy megadhatjuk az origó koordinátáit, az X tengely szögének dőlését a vízszinteshez képest és az X és Y tengely kapcsolatát.



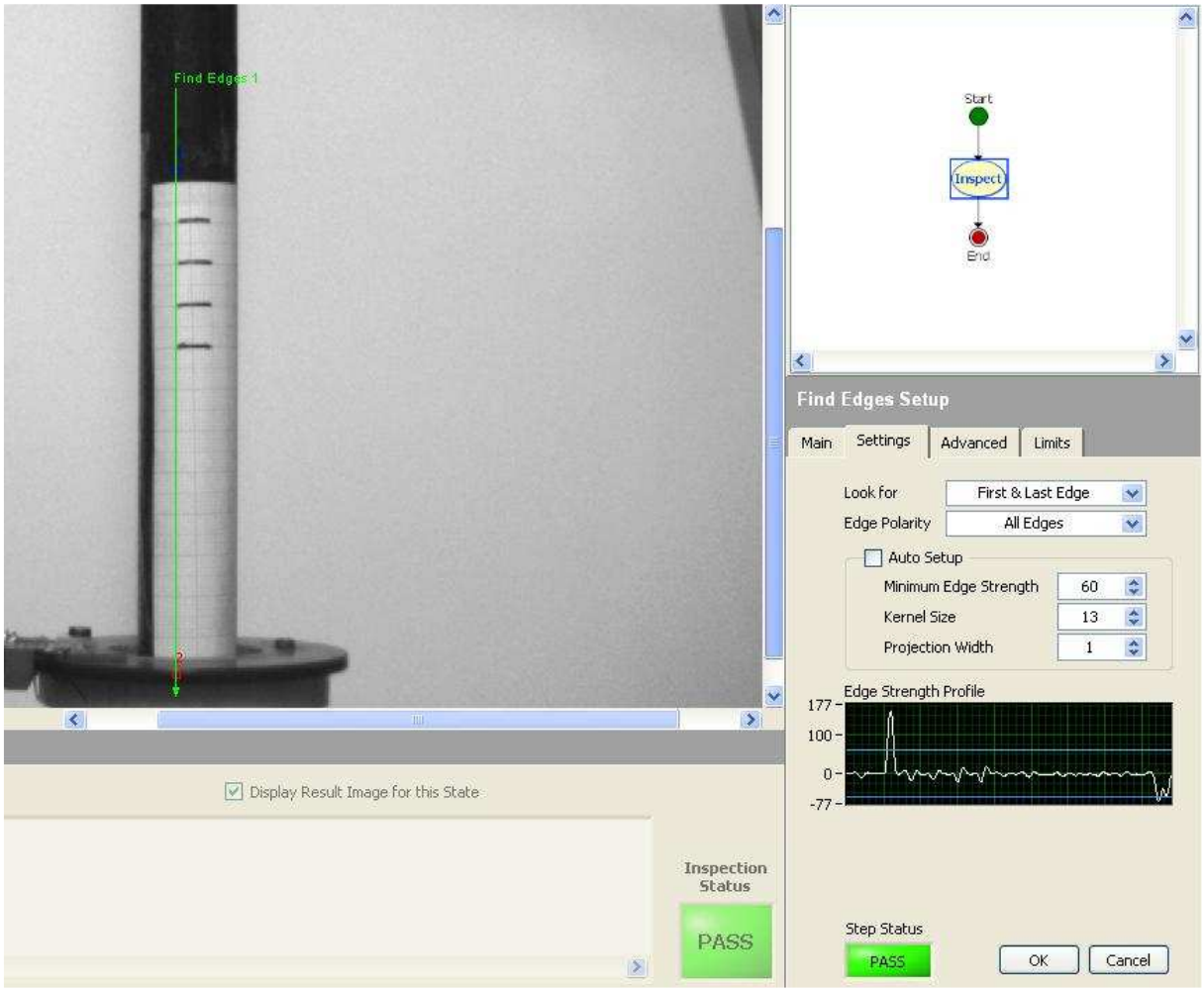
16. ábra Koordináta rendszer megadása

3.3.2. *Find Edge*



17. ábra Find Edge ikon

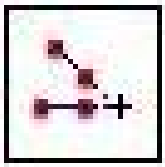
A Find Edge függvény mint ahogy a neve is elárulja éleket fog keresni az általunk megadott képen. A képen az egér segítségével húzunk egy vonalat, ezen az egyenes mentén fogja vizsgálni a függvény hogy van-e él.



18. ábra Find Edge

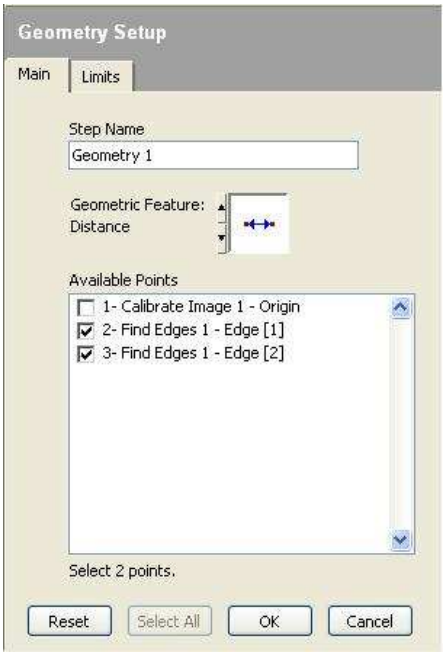
A 18. ábrán látható hogy több féle beállítási mód is lehetséges. A „Look for” vezérlővel lehet kiválasztani hogy az összes élt találja e meg, vagy csak az elsőt, esetleg az elsőt és utolsót vagy a legjobbat ami jelen esetben azt jelenti hogy a legélesebbet. Alatta az „Edge polarity” vezérlővel lehet beállítani, hogy csak a sötétebből a világosabba vagy csak a világosabból a sötétebbe átmenő éleket találja meg, és természetesen az is lehetséges, hogy az összes élt megtalálja. További beállítások is lehetségesek mint például, hogy minimum és maximum hány élet találjon és hogy egy élnél a pixel erőssége mennyire legyen kicsi vagy esetleg nagy.

3.3.3. Geometry



19. ábra Geometry ikon

Ez a függvény igen sokrétű. Pontokkal való számításokat végez, meg tudja adni például két pont által alkotott szakasz felezőpontját vagy megadhatja azt is, hogy erre a pontra állított merőlegesen van-e egy bizonyos, megadott harmadik pont. Ha éppen arra van szükségünk akkor a két pont által meghatározott, a merőlegessel vagy függőlegessel bezárt szöget is megadja, ez mind beállítás kérdése.

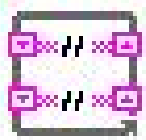


20. ábra Beállítások 1.

21. ábra Beállítások 2.

A „Geometric Feature” vezérlővel választhatunk ezen lehetőségek közül. A szakdolgozat tárgyát képező kísérlet esetén nekünk egyszerűen csak a két pont távolságára volt szükségünk. Ha esetleg több pontról van szó ki tudjuk választani melyik két pont távolságát szeretnék kiszámoltatni. Átkattintva a „Limits” fülre pedig láthatjuk hogy megadhatunk minimális és maximális távolságot is, és a már beállított két pont távolságát is kiírja (Distance: 12,35cm).

3.4.A JKI State Machine

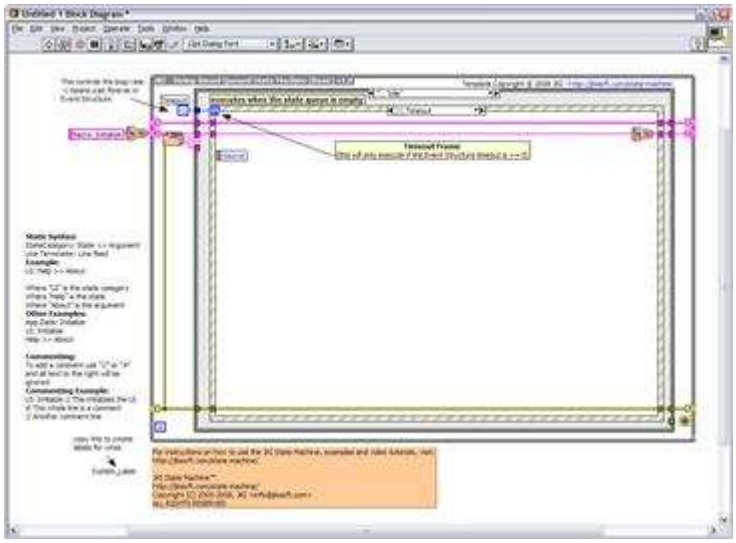


22. ábra State Machine ikon

[6]. A programozási feladat megoldását segítendő alkalmaztam még a JKI (James Kring, Inc.) fejlesztő csapata által kifejlesztett állapot-gépet, a State Machine-t.

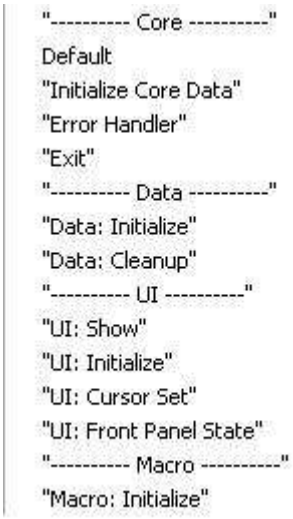
A State Machine a LabVIEW egy könnyen használható kiegészítője, ami alapját a sor adatszerkezet adja, melyek elemi karaktersorozatok.
A programban állapotváltozást előidéző eseményvezérlés valósul meg. Ezeket az állapotokat kiemelve és kategorizálva állt elő a program fő váza. A State Machine használatára azért került sor, mert a tapasztalatok szerint az eseményvezérlés kezelésére szánt idő közel a felére redukálódik, és nem tapasztalható program fagyás, ami elég gyakori egy felületesen összeállított esemény vezérelt struktúra használta esetén.
Rendkívül átláthatóvá vált a Blokk diagram, és egy teljesen új oldaláról segített felfedezni a LabVIEW-ban rejlő lehetőségeket. Az argumentumok átadására is lehetőség van minden különösebb programozás-technikai eszköz nélkül.

A State Machine használatához le kell tölteni a VI Package Manager-t, mely segítségével könnyedén letölthető sokféle kiegészítő modul, többek között a jki_lib_state_machine-2.0.0-1 mely szükséges a működéséhez.



23. ábra State Machine váza

A 23. ábrán láthatjuk az állapotgép üres vázát. Az állapotgépet két részre lehet elkülöníteni, az eseménykezelési keret és a végrehajtási keret. Az eseménykezelés során a felhasználói eseményeket kezeljük le, majd egy karaktersorozat megadásával hozzárendelhetjük, hogy melyik végrehajtási keret kerüljön futásra. A végrehajtási keretek további alkeretei az állapot keretek (state).



A mellékelt ábrán a kategóriák:

- Core
- Data
- UI
- Application

Végrehajtó keretek:

UI: Initializer

UI: Cursor Set

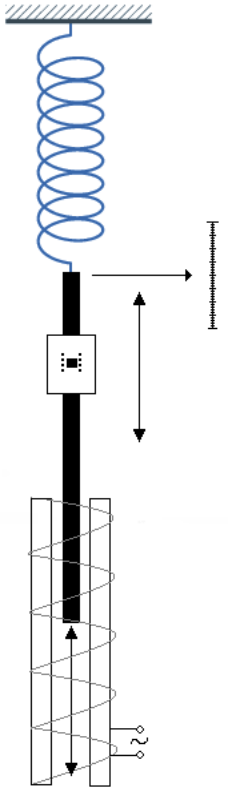
Tetszőleges kategóriákat hozhatunk létre, így egy esemény kezeléséhez számos állapot keretet rendelhetünk.

4. A MÉRÉS

4.1. Fizikai háttér

Debreceni Egyetem Természettudományi és Technológiai Kar Szilárdtest Fizika Tanszékén gyorsulásmérő eszközt építettek, mellyel egy rugóra szerelt vasrúd mozgását vizsgálták. A vasrúd csillapítatlan rezgőmozgást végez annak hatására, hogy a körülötte lévő tekercsbe váltóáramot vezettek, ami mágneses mezőt indukál benne. A mozgás amplitudója függ a váltóáram feszültségétől és frekvenciájától. A gyorsulás mérésére készített eszköz a rúd gyorsulását méri.

A mérés sikeressége érdekében a fekete vasrúdon tőle elütő – fehér - színű papírt helyeztünk el azért, hogy az erre a célra készített subVI az él keresés során biztosan felismerje a különböző pixeleket.



24. ábra A szerkezet felépítése

4.2.A szoftver célja

A szoftver célja, hogy egy gyorsulásmérő eszközt hitelesítsen. A mérés során megmérjük egy rugóra akasztott vasrúd gyorsulását a gyorsulásmérővel, majd az adatokat rögzítjük.

A létrehozott program egy kamera segítségével digitalizált képsorozat minden képét megvizsgálja. A képeken élek keresésével pontok kerülnek meghatározásra. A pixelben meghatározott pontok távolsága átváltásra kerül a hosszúság SI mértékegységének egyik prefixumára – centiméterre. A pontok így meghatározott távolsági adatának felhasználásával és a periódus idő segítségével, a képlet alapján a gyorsulás kiszámítható.

Ezt, a program által számolt adatot és a gyorsulásmérővel mért adatot összevetjük és amennyiben nincs szignifikáns eltérés a mérő hitelesítése eredményes.

4.3.A program működtetése

A program megjelenése egyszerű, áttekinthető.

A program indításakor az első lapon lehetőségünk van a számítógéphez csatlakoztatott kamera segítségével felvenni a kamera által közvetített képet, melyet elmentve .avi kiterjesztésű fájlt kapunk.

Amennyiben már rendelkezünk megfelelő képeket tartalmazó előzetesen rögzített mozgóképpel ezt a lépést nem kell elvégeznünk.

A második lapon látható a kép melyet kalibrálunk. Ezen a képen adjuk meg azt az egyenest mely egyenesen a program az éleket fogja keresni. A fehérről feketére vagy a feketéről fehérré váltásokat észleli a függvény, ezért is van a vasrúdra egy fehér papír ragasztva. A pontok kijelölését pontosan kell elvégezni annak érdekében, hogy a program sikeresen tudjon lefutni. Amennyiben a pontok nem úgy kerülnek kijelölésre, hogy a fekete-fehér átmenet élesen megkülönböztethető legyen előfordulhat, hogy az éleket nem találja meg a kereső és a pontok meghatározása nem lesz sikeres.

Ha a kalibrálást megfelelően beállítjuk, aztán megnyomjuk a „Make Measurement” gombot akkor a program megtalálja a két élváltás helyét (két pont). A méréshez a kiinduló adatok ezzel rendelkezésünkre állnak. A program beépített mechanizmusa elvégzi a szükséges műveleteket és kiírja a „Distance” nevű indikátorba a távolságát ennek a két pontnak - centiméterben.

A harmadik lap megjelenésében hasonló a másodikhoz. Itt a „Start Measurement” gombra kattintva felugrik egy ablak, ahol ki kell választanunk a videót melyet vizsgálni szeretnénk. A kiválasztás tallózással történik, mely után automatikusan elindul a lejátszás. A képernyőn figyelemmel kísérhetjük a program működését, látható, hogy a videó minden képkockáján folyamatosan kalibrálja és méri a távolságot a program.

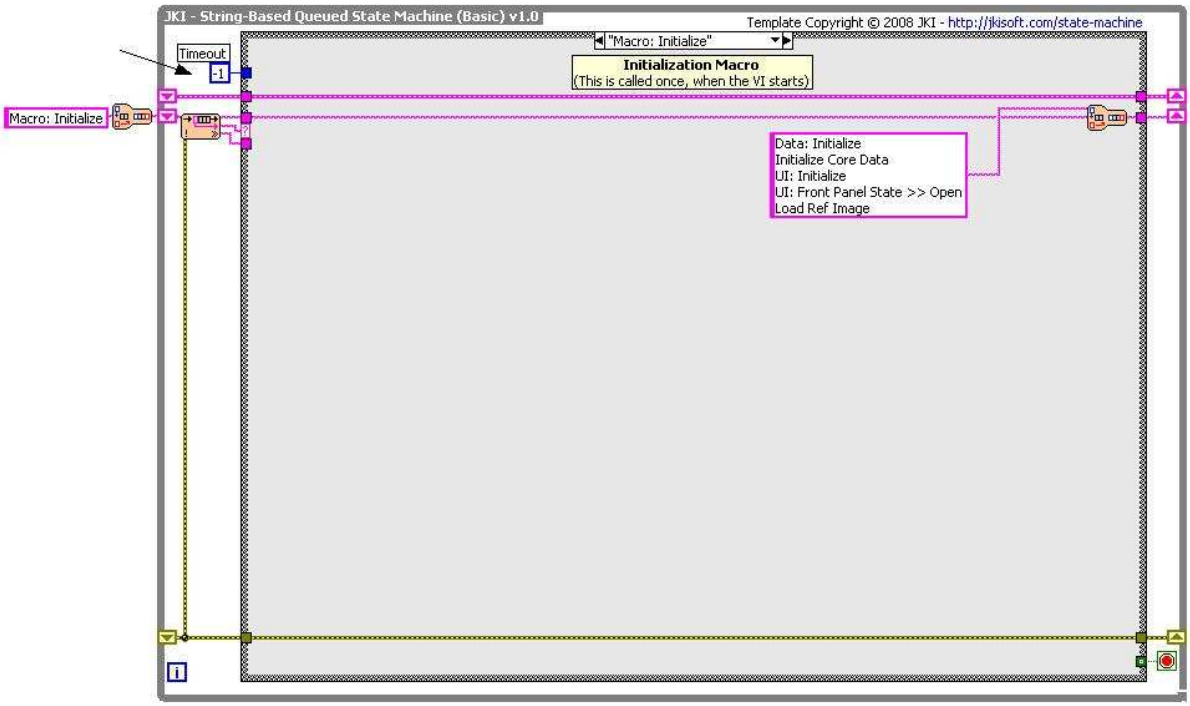
Ha a videónak vége átkattintva a negyedik oldalra látjuk az egyes mérésekből kiszámolt periódusidőt,maximális kitérést és gyorsulást.

Ugyanezen az oldalon a program grafikonos megjelenésben mutatja meg,

- az egyik grafikonon a kitéréseket
- a másik grafikonon a kitérések deriváltját.

4.4.A program felépítése

Mint ahogy már említettem a program alapját a State Machine adja. A State Machine-ben először a „Macro: Initialize” állapot keret fut le. Ebben – az alábbi képen is látható módon - egy stringben vannak megadva az állapotok nevei melyeket szeretnénk lefuttatni. Amilyen sorrendben vannak beleírva az állapotok olyan sorrendben fognak végrehajtódni.



25. ábra Macro: Initialize

A „Data: Initialize” részben adhatjuk meg a változókat melyeket összecsomagolunk (Bundle). Ezek közül bármelyik adatot kinyerhetjük bármelyik állapotban egy egyszerű név szerinti kicsomagolással (Unbundle by name).

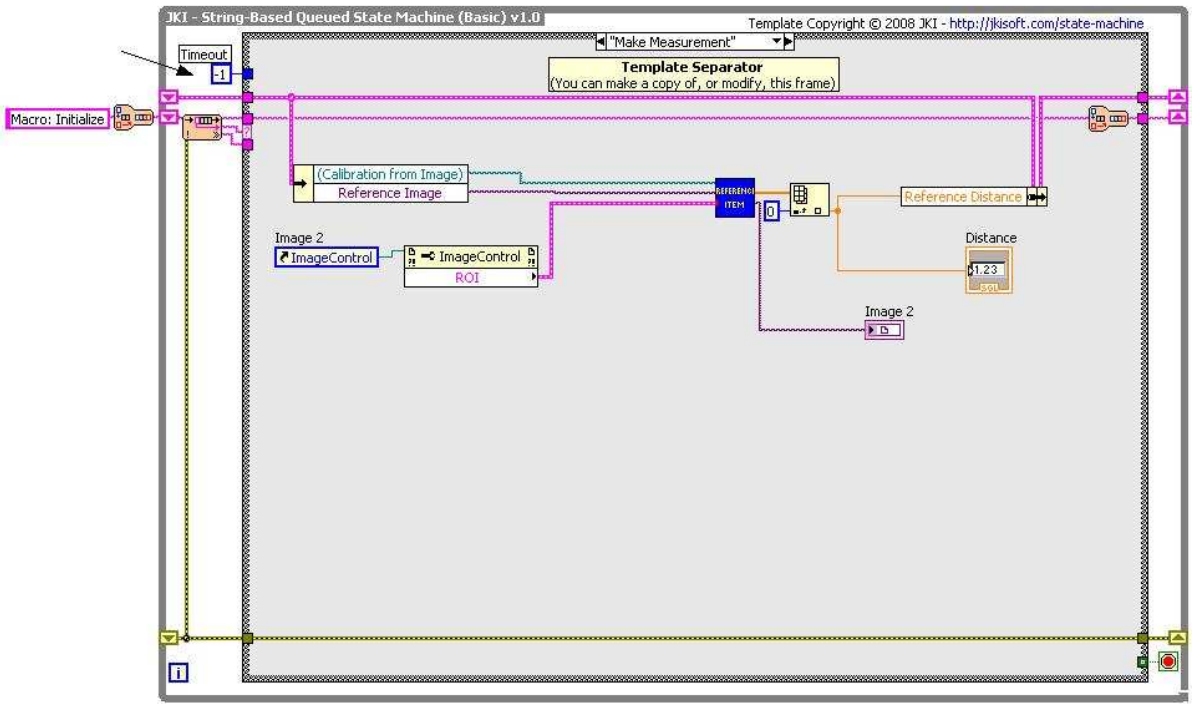
26. ábra Unbundle by name

A „Load Ref Image” már egy általam definiált állapot, melyben egy referencia képet nyittatunk meg a programmal. Amint ezek az állapotok lefutottak a program alvó, úgynevezett „Idle” állapotba kerül. Ebben az állapotban automatikusan benne van egy „Event

Structure”. Ez adja nekünk azt a lehetőséget, hogy ha megnyomunk egy gombot akkor lefusson az a programrész amire szükségünk van. Minden egyes gombhoz hozzárendelhetünk egy esetet ami lefut ha a gomb értéke (igaz vagy hamis) változik.

Több lehetőség is van, jelen esetekben én csak azt alkalmaztam mikor értékváltozásra fut le a megadott programrészlet.

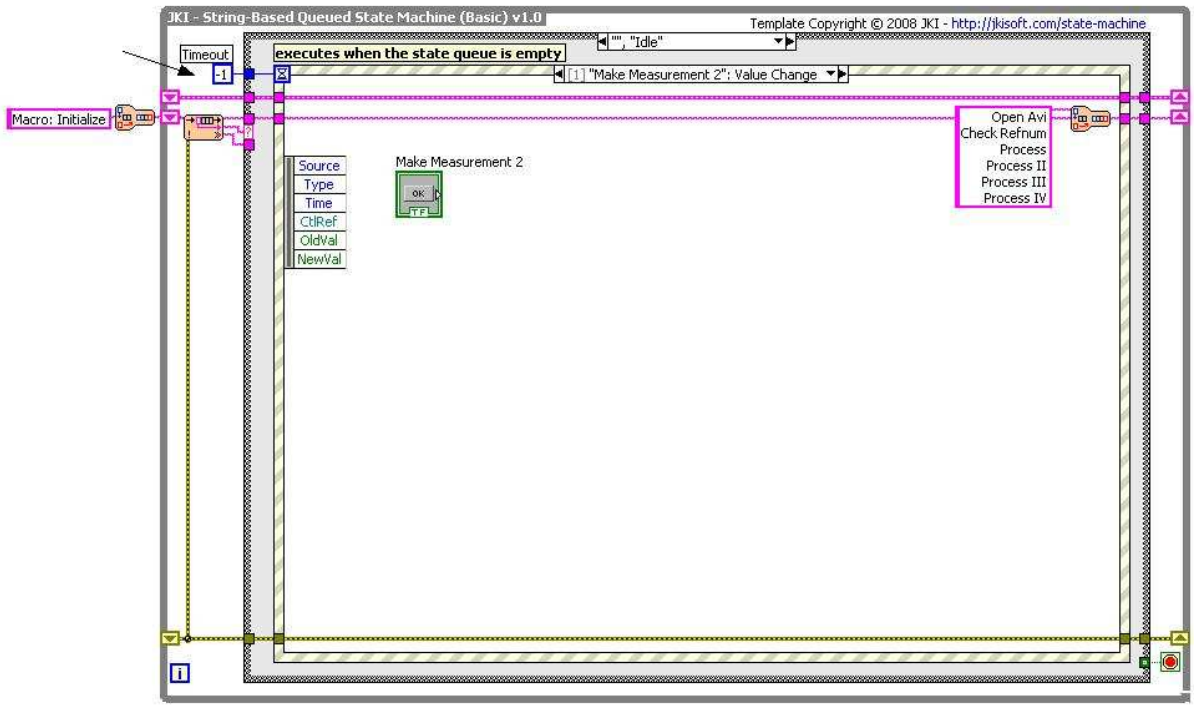
A „Make Measurement” gomb nyomására, mely a második lapon található, egy új string kerül a sorba, és lefuttatja a megfelelő programrészletet. Ebben a részben hívjuk meg először a subVI-t (Reference Item.vi) mely segítségével meghatározzuk a vasrúdon lévő fehér papír távolságát.



27. ábra Make Measurement

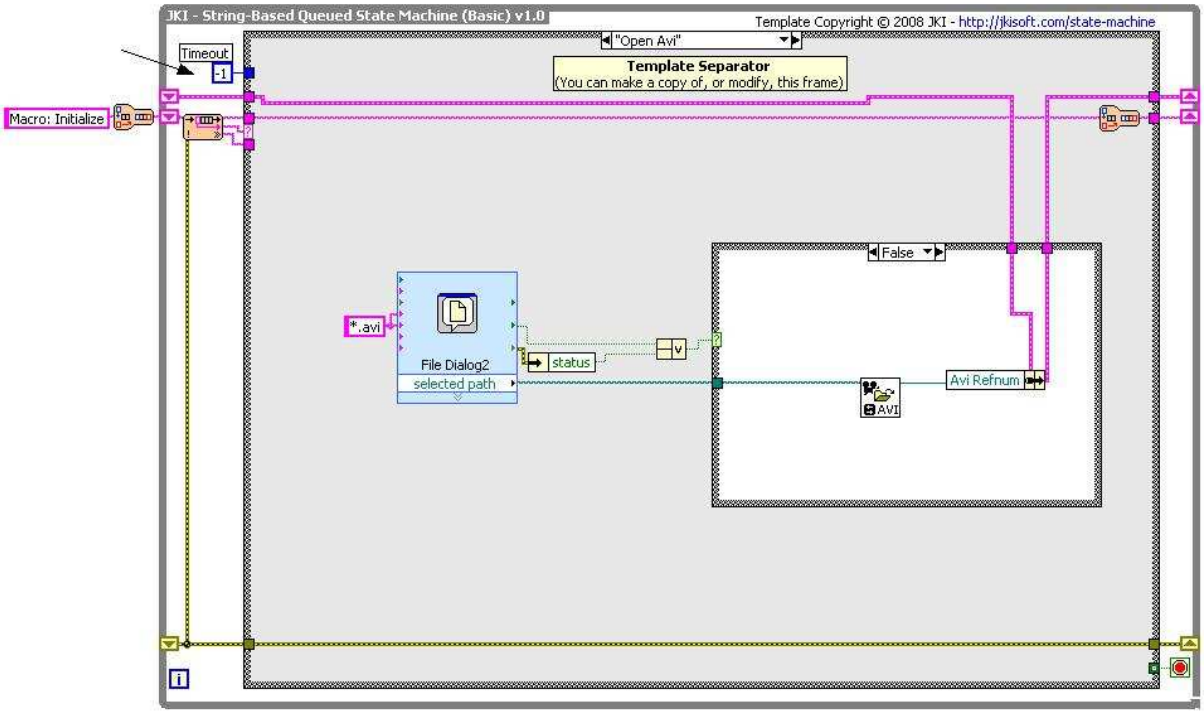
Ezt a subVI-t a Vision Builder AI programmal hoztam létre Ez a program vbai kiterjesztésű fájlokat hoz létre, de lehetőség van arra, hogy LabVIEW kódot (vi kiterjesztésű fájlt) generáljunk a programból. Az így generált kódot kicsit módosítva kaptam a Reference Item.vi-t.

A következő lapon a videót tudjuk kiválasztani a „Make Measurement” gombbal. A gomb - mint az előző lapon is - ha változik az értéke elindítja az „Event Structure”-ban a hozzátartozó, beállított programrész futását.



28. ábra Make Measurement 2

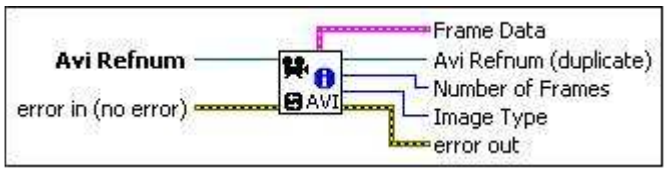
Mint látható a 28. ábrán itt több állapotot adunk át, melyeket le kell futtatnia a State Machine-nek. Az első az „Open Avi” állapot melyben az általunk kitallózott avi kiterjesztésű videót nyitja meg nekünk a programrészlet.



29. ábra Open Avi

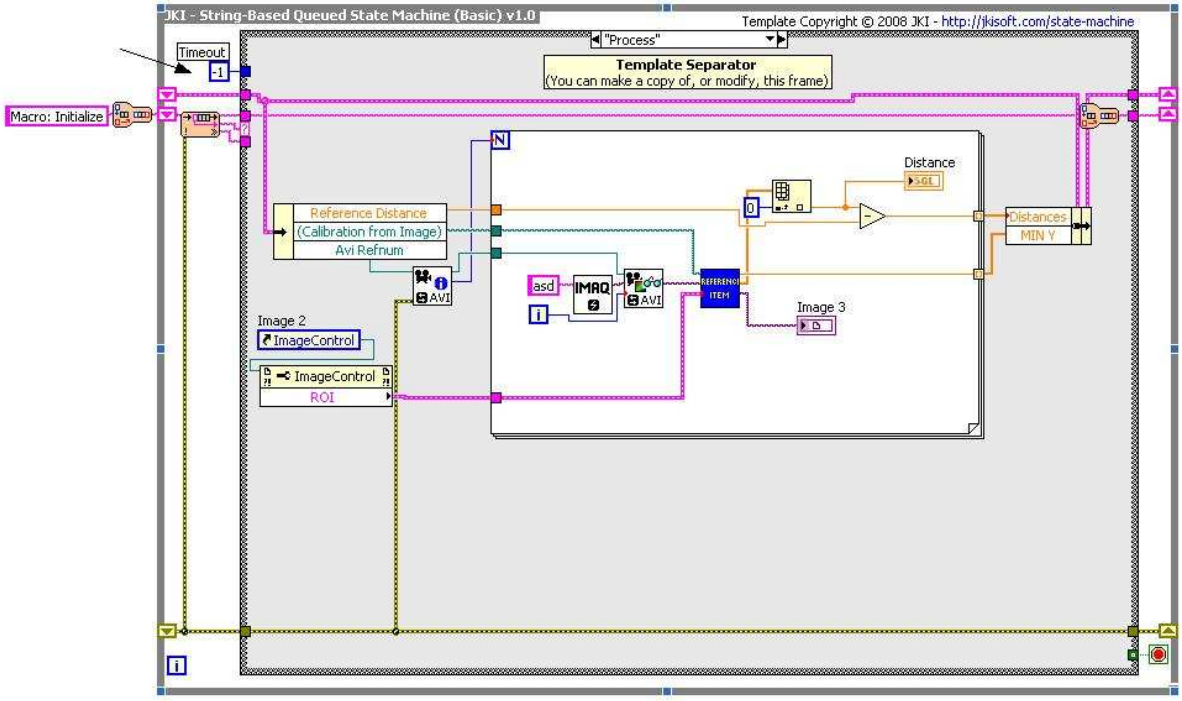
A megnyitott videót hozzákötjük (Bundle) az adatfolyamunkhoz melyet előre definiáltunk a „Data: Initialize” részben.

A következő futásra kerülő állapot a „Check Refnum” nevű állapot melyben ha a megnyitott videónk hibás vagy valamilyen probléma lépett fel a megnyitáskor ezt a hibát törli és újra meghívja az „Open Avi” állapotot. A video ellenőrzésre az IMAQ AVI Get Info VI-t használtam mely a videó fájlról ad meg nekünk információkat.



30. ábra IMAQ AVI Get Info VI

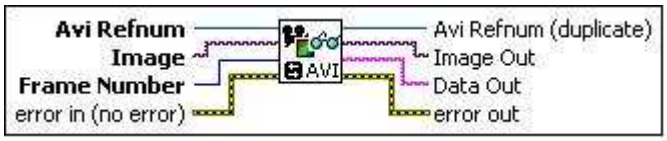
Ennek „Error Out” kimeneteléhez kötöttem egy Clear Errors VI-t, amely törli a hibát. Ha nem lépett fel hiba akkor tovább megy és meghívja a „Process” nevű állapotot.



31. ábra Process

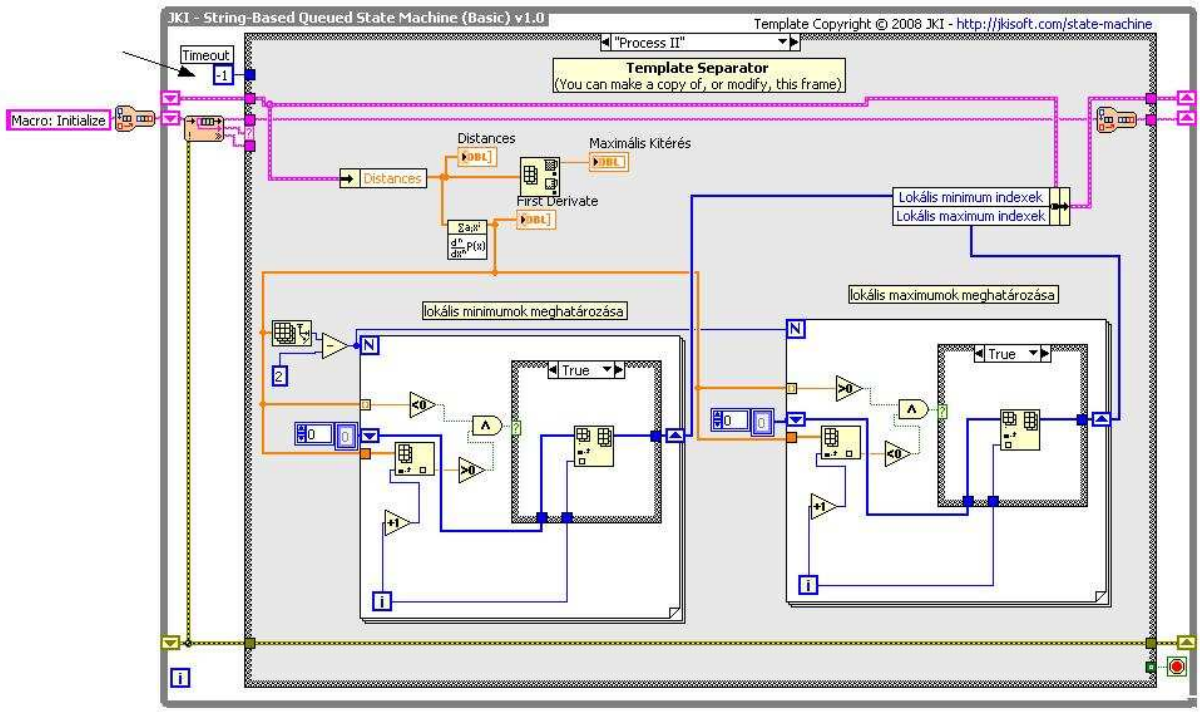
Ahogy a 31. ábra mutatja, ebben a részben a videó minden egyes frame-jére (köznapi neve „filmkocka”, angolul frame) meghívjuk a már említett Reference Item.vi-t, hogy minden

egyes frame-en mérje meg a vasrúdon látható fehér papír és tekercs távolságát. Emellett alkalmaztam még a IMAQ AVI Read Frame VI-t melynek megadhatjuk, hogy a videónk hányadik frame-jével szeretnénk dolgozni (Frame Number). Ezt hozzákötöttem a ciklus számlálóhoz mely megadja hogy hányadsorra fut le a ciklus. Ezáltal minden frame-et vizsgálva léphet tovább a program. Azt pedig hogy hányszor fusson le a ciklus az IMAQ AVI Get Info VI-lal adtam meg, ami visszaadja az adott videó frame számát a Number of Frames kimenetén. Így az IMAQ AVI Read Frame VI az adott frame-ről készült képet visszaadja nekünk az Image Out kimenetén.



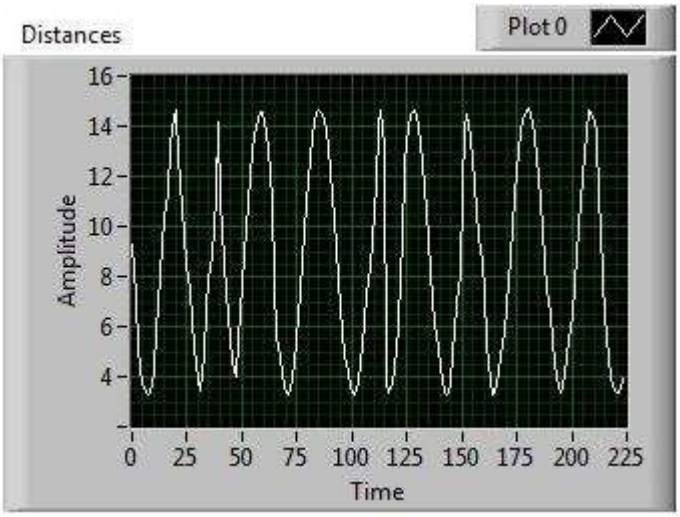
32. ábra IMAQ AVI Read Frame VI

Mikor a ciklus végigfutott a frame-eken mért távolságokat megkapjuk egy egy dimenziós tömbben. Ezt hozzákötjük az adatfolyamunkhoz és meghívódik a következő állapot, a „Process II”. Itt kicsomagolva az adatfolyamból az előbb említett tömböt, felhasználom, hogy megtudjam melyik frame-eken találhatóak az alsó illetve felső maximális kitérések, amplitúdók.



33. ábra Process 2

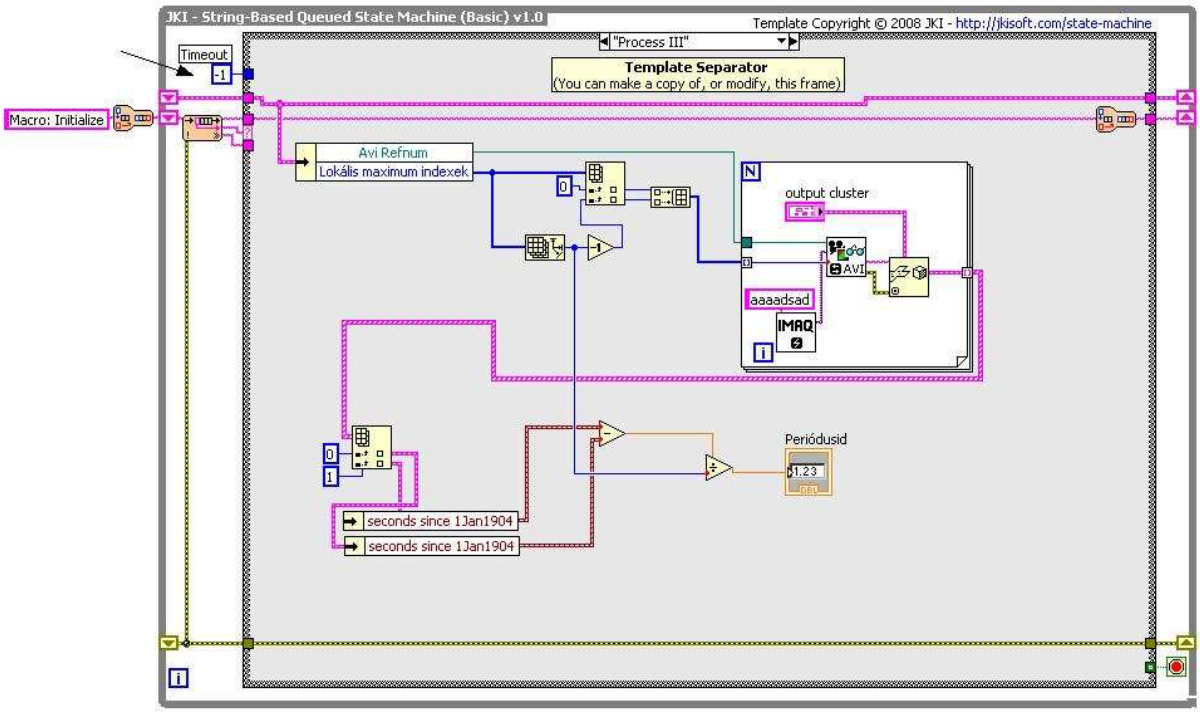
Először is veszem a tömb minden elemének az első deriváltját. A távolságok tömbjét és az első deriváltak tömbjét grafikonon láthatjuk a négyes oldalon.



34. ábra Távolságok grafikon

A grafikonon szabályos szinusz függvényt kéne látnunk, de az általam felvett videó magas minősége miatt a feldolgozás rendkívül nagy teljesítményű gépet igényelne. Így ahogy a grafikon is mutatja nem szabályos szinusz függvényt láthatunk. A tömbben annyi érték van ahány frame-ből épül fel a videó. A kapott tömb elemeit, melyek már a távolságok deriváltjai, összehasonlítom egymással, de csak a szomszédosokat. A derivált lényegében annak a mértéke, hogy egy egyváltozós valós függvény görbéjéhez rajzolt érintője milyen meredek. Ha emelkedő fázisban van a függvény görbéje akkor pozitívak lesznek ezek a számok, ha csökkenő fázisban akkor negatívak. Ezt fel lehet használni úgy hogy megnézem a szomszédos számok előjelét. Ha különböző előjelűek akkor ott váltás történt, tehát a görbe emelkedő fázisból csökkenőbe változott (ezek lesznek a lokális maximumok) vagy csökkenő fázisból emelkedőbe (ezek lesznek a lokális minimumok) vagyis ezen a ponton van egy maximális kitérés. Ekkor a két ciklusszámláló értékét lementjük egy-egy tömbbe, ez a videó egy-egy frame-jét jelenti. Így tehát megkaptuk a lokális maximum és minimum indexek tömbjét melyeket hozzákötünk az adatfolyamhoz.

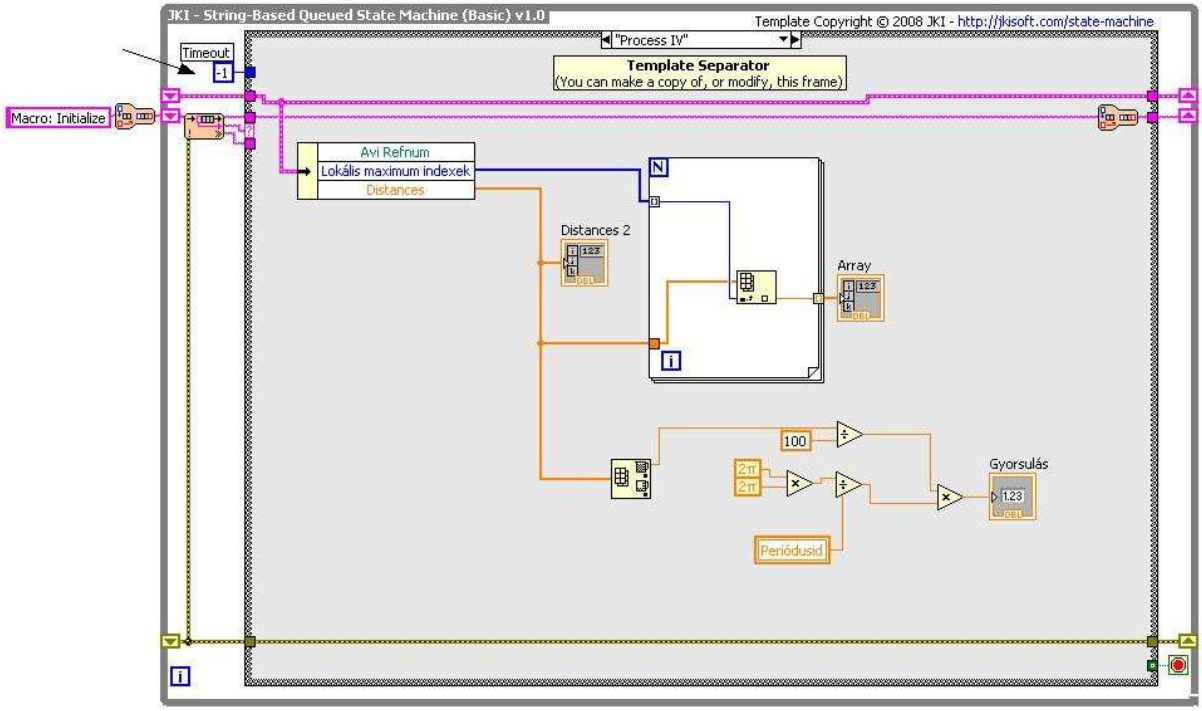
A következő állapot mely futásra kerül a „Process III”. Itt határozom meg a periódusidőt.



35. ábra Process 3

A periódusidő két azonos állapot között eltelt idő. A lokális maximum indexek tömböt kicsomagolva kinyerem belőle az első és utolsó elemet, ezeket felhasználva és a már említett IMAQ AVI Read Frame VI-al meghatározom a frame-eket melyekből információt szeretnék kinyerni. Ezek a frame-ek nem csak magát a képet tartalmazzák, hanem egyéb információkat is, mint például az időt. A két frame különböző időt tartalmaz, ezeket kivonva egymásból megkapjuk a két frame között eltelt időt. Ha a lokális maximum indexek tömb elemeinek a számát elosztjuk ezzel az idővel megkapjuk a periódusidőt, mivel az első és utolsó maximális kitérést vettem alapul ezért kell az összes elemszámmal elosztani.

A következő és egyben utolsó állapot amelyet beleraktunk a sorba a gombnyomással a „Process IV” nevű állapot.



36. ábra Process 4

A 37. ábrán jól látható, hogy ebben az állapotban egy egyszerű képlettel számoljuk ki a gyorsulást. Ehhez mindössze annyi kell hogy kicsomagoljuk az adatfolyamból a távolságok tömböt melyek közül a legnagyobbat elosztjuk százzal mivel méterrel kell számolnunk, aztán összeszorozzuk 2π/periódusidővel.

A Process 4 állapot lefutása után a program újra Idle állapotba kerül, miközben a negyedik lapon az eredményeket láthatjuk.

4.5.Mérési eredmények

A mérésekre a Debreceni Egyetem Természettudományi és Technológiai Kar Szilárdtest Fizika Tanszékén került sor.

A gyorsulásmérő által adott adat felhasználásával – figyelembe véve a gravitációs gyorsulás Magyarországon érvényes 9,81-m/s^2 értékét - meghatározható a gyorsulás.

A gyorsulásmérő alkalmazásával az alábbi mérési eredmények születtek:

	Frekvencia Hz	Feszültség V	Amplitúdó cm	Gyorsulás m/s^2
Mérés 1.	498,7	2	6,59	2,59
Mérés 2	498,7	1,6	5,24	2,06
Mérés 3	498,7	1,2	4,22	1,66
Mérés 4	475	1,5	6,46	2,55
Mérés 5	475	1,0	4,701	1,85
Mérés 6	475	0,5	1,94	0,76

1. Táblázat

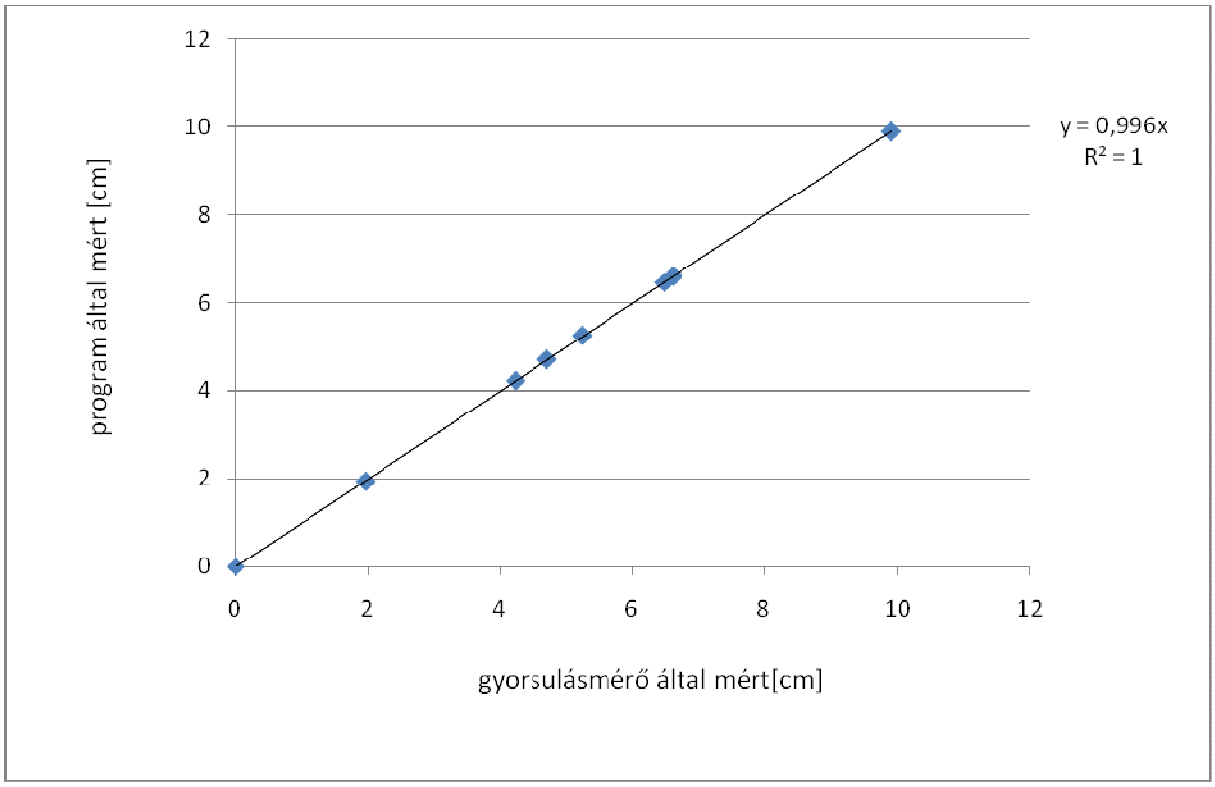
A programmal történő mérés alapján az alábbi adatok számolhatóak:

	Frekvencia Hz	Feszültség V	Amplitúdó cm	Gyorsulás m/s^2
Mérés 1.	498,7	2	6,62	2,61
Mérés 2	498,7	1,6	5,25	2,07
Mérés 3	498,7	1,2	4,23	1,66
Mérés 4	475	1,5	6,48	2,55
Mérés 5	475	1,0	4,71	1,85
Mérés 6	475	0,5	1,97	0,77

2. Táblázat

A táblázatok adatiból látható, hogy nincs lényeges eltérés a gyorsulásmérő és a programmal történő mérés adatai között.

A gyorsulásmérő eredményei és a programmal történő mérés alapján a kitéréseket páronként ábrázolom egy grafikonon. Itt is jól látható, hogy az eltérés elhanyagolható.



37. ábra Diagram

Mivel hasonlóak a kitérések ugyanazon periódusidő mellett, szintén csekély eltéréseket tapasztalunk a kiszámított gyorsulások között.

5. ÖSSZEFOGLALÁS

Összefoglalóan megállapíthatjuk, hogy a gyorsulásmérővel elvégzett mérések eredményeit az elkészített programból nyert adatok visszaigazolták, a gyorsulásmérő hitelesítése megtörtént. Az elkészített program a LabVIEW programnyelv és Vsion Builder AI kiegészítő programjainak felhasználásával biztosított optimális programozási felületet. A programnyelv használata során a kiegészítő függvények alkalmazása rendkívül előnyös lehetőségeket teremtett a mérések, a szükséges megfigyelések és számítások elvégzéséhez.

A program futásához input adatot szolgáltató videó felvétel megfelelő képfelbontással kellett, hogy elkészüljön annak érdekében, hogy a pixelek közötti átmenetek határozottan megkülönböztethetőek legyenek. A jó minőségű felvétel hátrányaként jelentkezett, hogy a fájl mérete rendkívül nagy lett, mely megkövetelte az átlagosnál nagyobb teljesítménnyel rendelkező számítógép igénybevételét.

A szakdolgozat elkészítéséhez szükséges programozási feladat teljesítéséhez alkalmazott programnyelvek és a kiegészítő függvények használata sok új ismeretet nyújtott. A program elkészítése során több függvényt, VI-t alkalmaztam melyek a program vázát jelentő State Machine használatával biztosították, hogy átlátható, jól kezelhető, könnyen javítható kódot hozzak létre. Az egyetemi tanulmányok alatt megismert programozási nyelvek közül számomra a legjobban használhatónak a LabVIEW programozási nyelv és a kiegészítő moduljai bizonyultak.

A program futása során bebizonyosodott, hogy a State Machine használata szükséges volt mert ez tette lehetővé - az előrejelzésnek megfelelően –, hogy a számítógép lefagyása nem hátráltatta a mérési feladatok elvégzését.

A képfeldolgozáson alapuló mozgásvizsgálat megvalósításához szükséges program elkészítéséhez a LabVIEW nyelv használata ideálisnak bizonyult hiszen ez egy olyan grafikus programozói nyelv amelynek legfőbb alkalmazási területe a mérési, tesztelési feladatok ellátása. Különösen hasznos lehetősége ezen nyelv használatának, hogy rendkívül szerteágazó

eszközkészlettel rendelkezik többek között az adatelemzés,megjelenítés, hibakeresés területén.

Ezen lehetőségeket is kihasználva sikerült olyan programot írni, amely az elvárásoknak megfelelt, a szükséges mérési és adatfeldolgozási feladatokat elvégezve a Debreceni Egyetem Természettudományi és Technológiai Kar Szilárdtest Fizika Tanszékén folyó kutatás eredményességéhez hozzájárult.

6. Irodalomjegyzék

1. A KÉPFELDOLGOZÁSRÓL
   http://www.mora.u-szeged.hu/~etel/digitalizalas/
   2009.11.17
2. A LabVIEW programozási nyelv
   http://en.wikipedia.org/wiki/LabVIEW
   2009.10.20
3. Icon and connector pane
   http://zone.ni.com/reference/en-XX/help/371361E-01/lvconcepts/icon_and_connector_pane/
   2009.10.24
4. Vision Builder AI
   http://digital.ni.com/manuals.nsf/websearch/3DF182BAE9365D82862575FB006B440C
   2009.11.09
5. A Vision Builder AI-ban használt főbb függvények
   http://zone.ni.com/reference/en-XX/help/372916H-01/
   2009.11.15
6. A JKI State Machine
   http://www.jkisoft.com/state-machine/docs/
   2009.11.12

7. Köszönet nyilvánítás

Köszönetet nyilvánítok a szakdolgozat elkészítéséhez nyújtott segítésért, valamint a tanszéki eszközök használatának engedélyezéséért témavezető tanáromnak Dr. Szabó István egyetemi docensnek, a Debreceni Egyetem Természettudományi és Technológiai Kar Szilárdtest Fizika Tanszék tanszékvezetőjének.

Köszönetemet fejezem ki Soha Rudolf Ferencnek a Debreceni Egyetem Természettudományi és Technológiai Kar Szilárdtest Fizika Tanszék munkatársának azért, hogy az általa végzett fizikai kísérlet részese lehettem, a szükséges eszközöket használhattam.

Köszönetemet fejezem ki Vámos Dánielnek a Debreceni Egyetem Természettudományi és Technológiai Kar Szilárdtest Fizika Tanszék munkatársának azért, mert segítette munkámat, a szükséges eszközök használatában segítségemre volt.